Design and Analysis of Algorithms (CSC311) – Spring 2017
Instructor: Prof. Mohamed Menai

Midterm Exam 1: Mar. 21, 2017 (5:00-6:30 PM).
Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.
Please, don't use pencils.

Student Name:.. Student ID:....................

Section:......................

This exam has 4 questions, for a total of 20 points.

Question 1 . . . . *4 points*

Prove or disprove that:

(a) [1 point] $n(n+1)/2 \in O(n^2)$

$\frac{n(n+1)}{2} = \frac{n^2+n}{2} \leq c \cdot n^2$

$\frac{n^2+n}{2} \leq \frac{2n^2}{2}$ $\quad c = 1 \quad n_0 = 1$

(b) [1 point] $n(n+1)/2 \in \Omega(n^2)$

$\frac{1}{2}(n^2+n) \geq c \cdot n^2$

$\frac{1}{2}n^2 + \frac{1}{2}n \geq \frac{1}{4}n^2$

$c = \frac{1}{4} \quad n_0 = 1$

(c) [1 point] $n(n+1)/2 \in \Theta(n^2)$

(d) [1 point] $n(n+1)/2 \in o(n^2)$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0 \ ?$$

$$\Rightarrow \lim_{n \to \infty} \frac{\frac{1}{2}n^2 + \frac{1}{2}n}{n^2} = \frac{1}{2} \neq 0$$

So .... $\notin o(n^2)$

Question 2 .... *6 points*

For each of the following functions, indicate the class $\Theta(g(n))$ the function belongs to. Use the simplest $g(n)$ possible in your answers.

(a) [2 points] $f(n) = (n^2+1)^{10}$

$O(g(n)) =$

$(n^2+1)^{10} \leq c_1 \, g(n)$

$\leq (n^2+n^2)^{10}$

$\leq 2^{10} n^{20}$

$c = 1024, \ n_0 = 1$ $\quad O(n^{20})$

$\Omega(g(n)): (n^2+1)^{10} \geq c_2 \, g(n)$

$\geq n^{20}$ $\quad n_0 = 1$

$c_2 = 1 \quad n_0 = 1 \quad \Omega(n^{20})$

(b) [2 points] $f(n) = 2n\log(n+2)^2 + (n+2)^2\log(\frac{n}{2})$

(c) [2 points] $f(n) = \sqrt{10n^2 + 7n + 3}$

$(10n^2+7n+3)^{\frac{1}{2}} \geq (10n^2 + 7n^2 + 3n^2)^{\frac{1}{2}}$

$\geq \sqrt{20n^2}$

$\geq \sqrt{20}\, n$

$c = \sqrt{20} \quad n_0 = 1$

$\leq \sqrt{10n^2}$

$\leq \sqrt{10}\, n$

$c_2 = \sqrt{10} \quad n_0 = 1$

Question 3 .... *4 points*

Consider the following algorithm.

Algorithm 1 Unknown($A[0..n-1]$) ▷ Input: An array $A[0..n-1]$ of n real numbers

$minval \leftarrow A[0]$ 1
$maxval \leftarrow A[0]$ 1
for $i \leftarrow 1, n-1$ **do** n-1
    if $A[i] < minval$ **then** n-2
        $minval \leftarrow A[i]$
    end if
    if $A[i] > maxval$ **then** n-2
        $maxval \leftarrow A[i]$
    end if
end for
return $maxval - minval$ 1

(a) [1 point] What does this algorithm compute?

it gives the diff betwen
max element and min element

(b) [1 point] What is its basic operation?

Comparision

(c) [2 points] Give the best-case and worst-case running times of this algorithm in asymptotic notation.

Question 4 . . . . *6 points*

Describe an algorithm that takes a list of n positive integers and finds the location of the last even integer in the list, and returns 0 if there are no even integers in the list. Give

Midterm I Exam, spring 2013 **Saturday March 19th, 2013** **Exam time: 07:00-9:00 P.M.**

Student's name:.. **ID:** .... **Section:** ....

Problem 1

(a) Give the following functions a number in order of increasing asymptotic growth rate. If two functions have the same asymptotic growth rate, give them the same number.

Function	Rank
$8^{\lg n} + 2n$ n^3	3
$5 \lg n^8$ $\lg n$	1
$n^3 + 5n^2 - 100$ n^3	3
$\frac{n^2}{\lg n}$ $n^2 \le \ \le n$	2
$2 \lg n + \lg(\lg n^2)$ $\lg n$	1
3^{2n} 3^n	4

(b) Using the definition of θ, find $g(n)$, C, and n_0 in the following:

$\Omega(g(n))$

$4n^6 - 2n + n \ge 3n^6$

$n_0 = \frac{3}{1} = 3$

$c = 3 \quad n_0 = ?$

$O(g(n))$

$4n^6 - 2n^2 + n \le c \cdot n^6$

$\le 7n^6$

$c = 7, n_0 = 1 \quad O(n^6)$

$$4n^6 - 2n^2 + n \in \theta(g(n))$$

Problem 2 (6 points)

Consider the following recurrence relation:

$$T(n) = 2T(\tfrac{n}{2}) + 2n.$$

(a) Solve this recurrence relation using recursive substitutions.

(b) Find g(n), where **T(n) ∈ O (g(n)).**

Hint: $\sum_{i=0}^{n-1} 2^i = 2^n - 1.$

$T(n) = 2T(\frac{n}{2}) + 2n$

$= 2[2T(\frac{n}{4}) + \frac{2n}{2}] + 2n$

$= 4T(\frac{n}{4}) + 2n + 2n$

$= 4[2T(\frac{n}{8}) + \frac{2n}{4}] + 2n + 2n$

$8T(\frac{n}{8}) + 2n + 2n + 2n$

Problem 3 (6 points)

Solve the following recurrences using the Master theorem by giving tight θ-notation bounds. Justify your answers.

(a) T(n) =8T(n/2) + 5n^2

(b) T(n) =9T(n/3) + 3n^2

(c) $T(n) = 7T(n/2) + n^3$

Problem 4

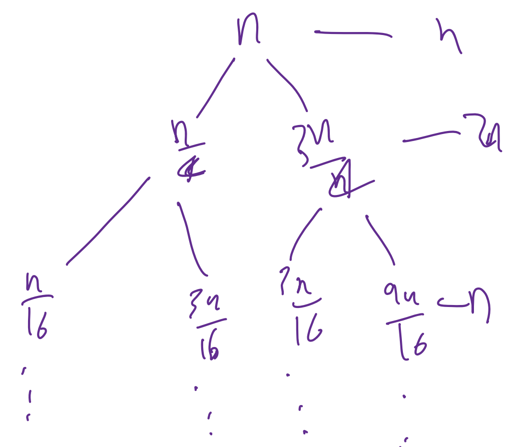
For each algorithm listed below, give a recurrence that describes its worst-case running time, and give its worst-case running time using O-notation.
You need **not** justify your answers.

(a) Merge sort n log n

(b) Insertion sort n^2

(c) Quicksort algorithm n log n

(d) Binary Search Log n



Problem 5 (4 points)

Consider a variation of MergeSort which divides the list of elements into two lists of size 1/4 and 3/4, recursively at each step, instead of dividing it into halves. The Merge procedure does not change.

(a) Give a recurrence relation for this algorithm $T(n) = T(\frac{n}{4}) + T(\frac{3n}{4}) + 2$
(b) Draw a recursion tree for the algorithm
(c) Using the recursion tree, explain how you can deduce the worst case upper bound.

Problem 6

Consider the problem below, and suggest **TWO** algorithm design techniques and give a high-level description of the **TWO** algorithms to solve it.

a- There are n closed boxes numbered from 1 to ***n*** and you are told that there are ***k*** balls in the first ***k*** boxes (one ball in each box), and all other boxes are empty. How to find the value of ***k***?

Master Theorem:

If **T(*n*) = *a* T(*n/b*) + f(*n*)** then

$$T(n) = \left\{ \begin{array}{ll} \Theta\left(n^{\log_b a}\right) & f(n) = O\left(n^{\log_b a - \varepsilon}\right) \\ \Theta\left(n^{\log_b a} \log n\right) & f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) \text{AND} \\ & af(n/b) < cf(n) \text{ for large } n \end{array} \right\} \begin{array}{l} \varepsilon > 0 \\ c < 1 \end{array}$$

Midterm II Exam, Spring 2013 **Monday April 29th, 2013** **Exam time: 07:00-9:00 P.M.**

Student's name:.. **ID:** .... **Section:** ....

Problem 1 (8 points)

For each of the question below, circle either T (for **True**) or F (for **False**). **No explanations** are needed. Incorrect answers or unanswered questions are worth zero points.

T **F** Given a graph $G = (V, E)$ with cost on edges and a set $S \subseteq V$, let (u, v) be an edge such that (u, v) is the minimum cost edge between any vertex in S and any vertex in V-S. Then, the minimum spanning tree of G must include the edge (u, v). (You may assume the costs on all edges are distinct, if needed).

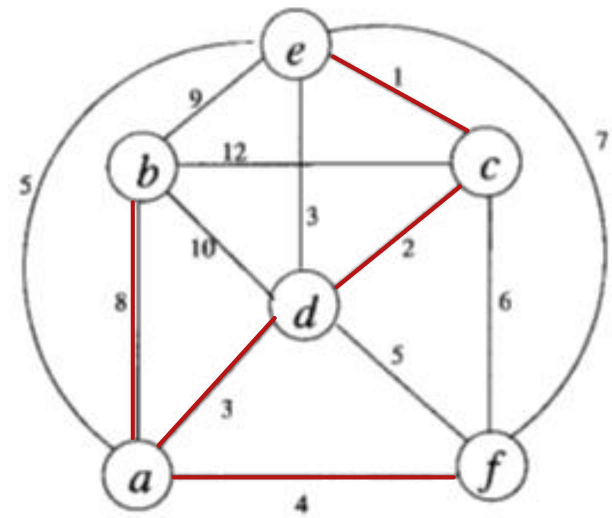
T **F** Let G be an edge-weighted directed graph with source vertex s and let T be a shortest path tree from s. Suppose we add a positive constant p to the cost of every edge in G. T remains a shortest path tree from s.

T **F** Let $G = (V, E)$ be a weighted graph and let M be a minimum spanning tree of G. The path in M between any pair of vertices v_1 and v_2 must be a shortest path in G

T **F** Consider a communication network of nodes where node v needs to broadcast a single message to all the other nodes efficiently. The message should be sent to the shortest paths tree from v.

Problem 2 (20 points)

For each of the algorithm below, list the edges of the Minimum Spanning Tree for the graph in the order selected by the algorithm.



a- Prim's algorithm starting at vertex a.

b- Kruskal algorithm.

Problem 3 (9 points)

Compare dynamic programming, greedy programming, and standard recursion by filling out the table below

Algorithm	*Top-down or bottom-up?*	*Solve the same subproblem once?*	*Always solve all sub-problems*
Dynamic Programming	Bottom-up	yes	yes
Greedy Programming	Top-down	no	no
Standard Recursion	Top-down	no	yes

Problem 4 (13 points)

Suppose **X**= cars and **Y**= cesar.

(a) Compute the length of an LCS of X and Y by filling out the c-matrix below

		c	e	s	a	r
c		1	1	1	1	1
a		1	1	1	2	2
r		1	1	1	2	3
s		1	1	2	2	3

c a r

(b) What is LCS of **X** and **Y**? For $\mathbf{X}_i$ and $\mathbf{Y}_j$ to be match in the LCS, what must be true about **c**[i,j]?

Car,

if X[i] = Y[j]
c[i,j]=C[i-1, j-1]+1

Problem 5 (10 points)

Give the pseudo-code of an algorithm that takes as input an array **A** of integers, and returns the length of the longest contiguous subsequence of odd numbers in **A**.

Example:

The length of the longest contiguous zeros subsequence in [1, 2, 19, 5, 4, 7, 51, 23, 22, 13, 15, 36] is 3.

1 0 1 2 0 1 2 3 0 1 2 0

What is the time complexity of your algorithm? hint: Use Dynamic programming paradigm.

```
LCO(A[1 ... n])
  S[ ] ← ∅

  for i → n do
      S[i] ← 0

  for i=1 to n do
     if(A[i] % 2 == 1)
       S[i] = S[i-1] + 1
     else
       S[i] = 0

  return max(S)
```

King Saud University

College of computer and Information Sciences

CSC 311 – Design and Analysis of Algorithms

جامعة الملك سعود

كلية علوم الحاسب والمعلومات

311 عال –

MidTerm Exam II, Fall 2018 | November 6th | Exam time: 17:00-18:30

Student's [redacted] ID [redacted] Section: ......11:00...AM

23.5/30

Problem 1 (2 points) 2/2

Compare dynamic programming and standard recursion by filling out the table below:

Algorithm	*Top-down or bottom-up?*	*Solve the same subproblem once?*
Dynamic Programming	bottom	yes
Standard recursion	top	no

Problem 2 (9 points) 8/9

9 8

Let X and Y be two strings such that X= "recursion" and Y= "election"

We define X_i and Y_j as two prefixes of X and Y of length i and j, respectively.

We use a matrix C to store the optimal solutions of the subproblems. Let $C[i, j]$ be the length of Longest Common Subsequence (LCS) of X_i and Y_j.

(a) For X_i and Y_j what must be true about $C[i, j]$?

$$C[i,j] = \begin{cases} C[i-1, j-1] & \text{if } x[i] = y[j] \\ \max\{C[i-1,j], C[i,j-1]\} \end{cases}$$

(b) Compute the length of an LCS of X and Y by filling out the matrix C.

(c) What is LCS of *X* and *Y*? Explain how you can obtain the LCS from the table *C*.

	Y_j	e (1)	l (2)	e (3)	c (4)	t (5)	i (6)	o (7)	n (8)
X_i	0	0	0	0	0	0	0	0	0
1 r	0	0	0	0	0	0	0	0	0
2 e	0	1	1	1	1	1	1	1	1
3 c	0	1	1	1	2	2	2	2	2
4 u	0	1	1	1	2	2	2	2	2
5 r	0	1	1	1	2	2	2	2	2
-1 6 s	0	1	1	1	2	2	2	2	2
7 i	0	1	1	1	2	2	3	3	3
8 o	0	1	1	1	2	2	3	4	4
9 n	0	1	1	1	2	2	3	4	5

e c i o n

The explain:

I use this formula when I fill up the table:

if $x_i = Y_j$

$c[i-1, j-1]+1$

otherwise

$\max(c[i-1,j], c[i,j-1])$.

After I fill up the table, I determine the length and I do like what I drew on the matrix.

∴ LCS of X and Y = ecion

Problem 3 (9 points) 8.5/9

Let $A_1, \ldots, A_4$ be matrices with dimensions 2×3, 3×10, 10×3, 3×5, respectively. In finding an optimal parenthesization of the matrix chain product $A_1 * A_2 * A_3 * A_4$, we use two tables **m[,]** and **s[,]** below. Here **m[i,j]** stores the optimal cost of computing subchain $A_i \ldots A_j$ and **s[i,j]** records the index k where the optimal parenthesization splits $A_i \ldots A_j$ between A_k and A_{k+1} for some k with $i \le k \le j-1$.

A_1 (2x3) . A_2 (3x10) . A_3 (10x3) . A_4 (3x5)

a- What is the recursive equation of **m[i,j]**?

if $i=j$, $m[i,j]=0$,
otherwise,

$m[i,j] = \min \{ m[i,k] + m[k+1, j] + d_{i-1} * d_k * d_j \}$

-0.5 $1 \le k <$

b- Fill the empty entries in the two tables. Show your work in each case.

m

j \ i	1	2	3	4
4				0
3			0	
2		0		
1	0			

s

j \ i	1	2	3
4			
3			0
2		0	

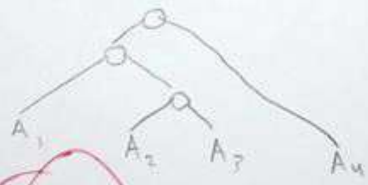
← My table in the back of previous page.

c- Now, give the optimal parenthesization of the matrix chain product *A1*A2*A3*A4*. Show how you came up with the solution using the tables above.

$((A_1)\cdot(A_2\cdot A_3))\cdot(A_4)$

First: I compute that M[1,4]=3, so I split on A_3 (form left to right)

second: M[1,3]=1, so I split on A_1 ⟶ $(A_1)\cdot(A_2 \cdot A_3)$



Problem 4 (5 points) 00/5

Consider the following equation system:

weight وزنة
value قيمة

$$\max \quad x_1 + 4x_2 + 3x_3$$

$$x_1 + 4x_2 + 3x_3 \quad \leq \quad 4$$

weight

$P_0 = 2 \quad P_1 = 3 \quad P_2 = 10 \quad P_3 = 3 \quad P_4 = 5$

i

S	1	2	3	4
j: 1	0	60	108	138
2		0	90	135
3			0	150
4				0

(i)

M	1	2	3	4
(j): 1		1	1	3
2			2	3
3				3
4				

$$\underset{2\times3}{A_1} \cdot \underset{3\times10}{A_2} \cdot \underset{10\times3}{A_3} \cdot \underset{3\times5}{A_4}$$

* $m[1,1] = 0$, $m[2,2] = 0$, $m[3,3] = 0$, $m[4,4] = 0$

$\boxed{*\ m[1,2]}$

$A_1 \cdot A_2 = 2\times3\times10 = 60$

$\boxed{*\ m[2,3]}$

$A_2 \cdot A_3 = 3\times10\times3 = 90$

$\boxed{*\ m[3,4]}$

$A_3 \cdot A_4 = 10\times3\times5 = 150$

$\boxed{*\ m[1,3]}$

$$\left.\begin{array}{c} (A_1 \cdot A_2) \cdot A_3 \\ m[1,2] + m[3,3] + 2*10*3 \\ 60 + 0 + 60 = 120 \end{array}\right\} \quad \begin{array}{c} A_1 \cdot (A_2 \cdot A_3) \\ m[1,1] + m[2,3] + 2*3*3 \\ 0 + 90 + 18 = \underline{108} \\ \text{minimum} \end{array}$$

$\boxed{*\ m[2,4]}$

$$\left.\begin{array}{c} (A_2 \cdot A_3) \cdot A_4 \\ m[2,3] + m[4,4] + 3*3*5 \\ 90 + 0 + 45 = 135 \\ \text{minimum} \end{array}\right\} \quad \begin{array}{c} A_2 \cdot (A_3 \cdot A_4) \\ m[2,2] + m[3,4] + 3*10*5 \\ 0 + 150 + 150 = 300 \end{array}$$

$\boxed{*\ m[1,4]}$

$$m[1,4] = \min\{ m[1,1] + m[2,4] + 2*3*5,\ m[1,2] + m[3,4] + 2*10*5,\ m[1,3] + m[4,4] + 2*3*5 \}$$

$$= \min\{ 0 + 135 + 30 = 165,\ 60 + 150 + 100 = 310,\ 108 + 0 + 30 = 138 \}$$

1 0 1 0 1 0 1

Problem 5 (5 points) 5/5

Give the pseudo-code of an algorithm that takes as input an array **A** of integers, and returns the length of the longest contiguous subsequence of odd numbers in **A**.

Example:

1 0 1 2 0 1 2 3 0 1 2 0

The length of the longest contiguous ~~zeros~~ odd subsequence in [1, 2, 19, 5, 4, 7, 51, 23, 22, 13, 15, 36] is 3.

What is the time complexity of your algorithm? hint: Use Dynamic programming paradigm.

4

```
int m = B[0]

for ( int i=1; i< length of B ; i++) {
    if ( B[i] > m)
        m = B[i] ;

}

return m ;
```

which is maximum number in the array

(max seq. of odd numbers)

Student's name: .... **ID:** .... **Section:** ....

Problem 1 (10 points)

For each of the question below, circle either T (for **True**) or F (for **False**). **No explanations** are needed. Incorrect answers or unanswered questions are worth zero points.

T F Consider a communication network of nodes where node v needs to broadcast a single message to all the other nodes efficiently. The message should be sent to the shortest paths tree from v.

T F Prim's algorithm is a greedy solution of the Minimum Spanning Tree problem.

~~T F Bellman-Ford solves the Minimum Spanning Tree problem~~.

T F Let T be the minimum spanning tree of a connected, undirected, and weighted graph G = (V, E). If e is a new edge, then $T \cup \{e\}$ contains a cycle.

T F Let G be an edge-weighted directed graph with source vertex s and let T be a shortest path tree from s. Suppose we add a positive constant p to the cost of every edge in G. T remains a shortest path tree from s.

T F Let $G = (V, E)$ be a weighted graph and let M be a minimum spanning tree of G. The path in M between any pair of vertices v_1 and v_2 must be a shortest path in G

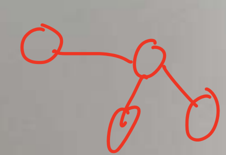
~~T F Bellman-Ford algorithm works on all graphs with negative-cost edges~~.

T F Dijkstra's algorithm works on graphs with negative-cost edges.

T F Let G be an edge-weighted directed graph with source vertex s, and let T be a shortest path tree from s. If we add a positive constant p to the cost of every edge incident on s in G. T remains a shortest path tree from s.

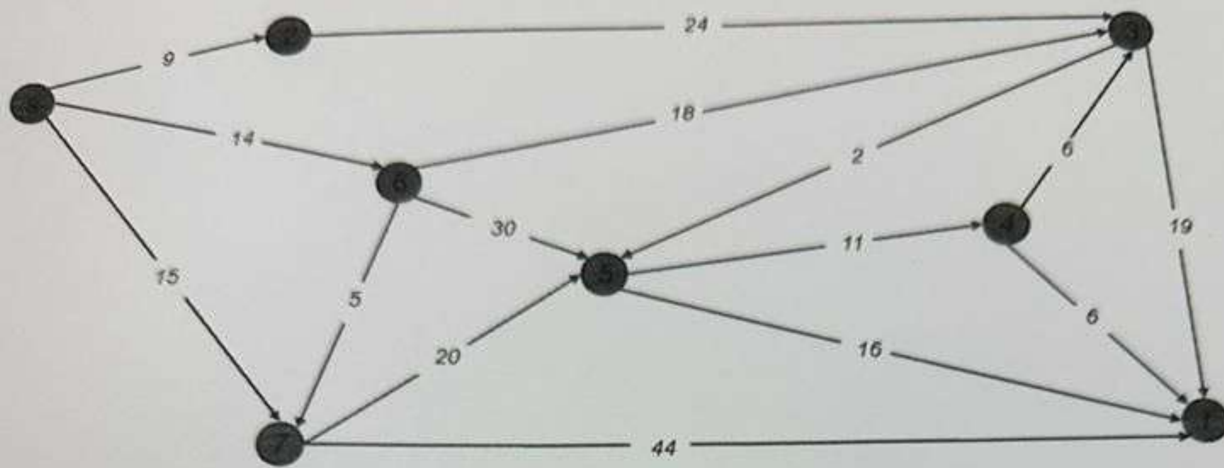
T F The running time of Dijkstra's algorithm is O(V+E) where V is the number of vertices and E is the number of edges,

VLogE



Problem 2 (6 points)

Apply Bellman Ford algorithm on the graph below to solve the single source shortest path starting from vertex S. You need to show the results of each step with all details (hint: use a table as we did in class). Also, you need to show the final results.



	s	2	3	4	5	6	t
d[i]	*0*	+∞	+∞	+∞	+∞	+∞	+∞
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

Problem 3 (6 points)

Use a graphical representation to show (step by step) how Prim's algorithm constructs the minimum spanning tree of the graph in Fig. 1. You need to show the subtree resulting from every step assuming that we start from the vertex v1.

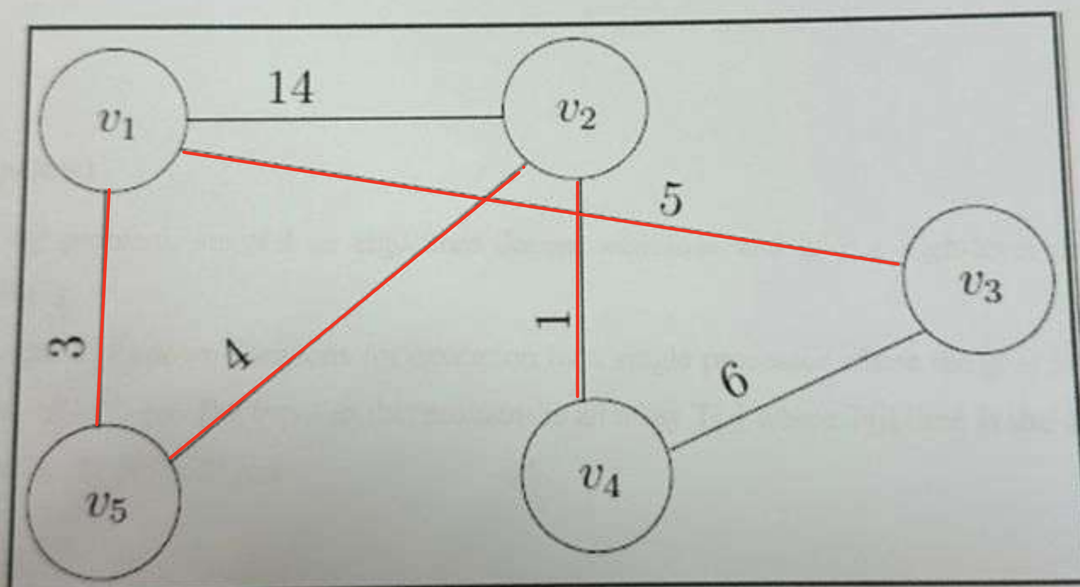


Figure 1: Undirected weighted graph.

Problem 4 (8 points)

For the following problem, suggest an algorithm design technique and give a high-level description of an algorithm to solve it.

(a) Scheduling n jobs of known durations for execution by a single processor where the goal is to minimize the total waiting time of all jobs. The input in this problem is an array T[ ] where T[i].time is the duration of the i^{th} job and T[i].id is the ID of the i^{th} job.

We'll use greedy design techinque

SJobs(T[1...n])

Sort T[i].time in ascending order
int time, wait ← 0
for i=1 to n do
time += T[i].time
wait += time − T[i].time

return wait

(b) Prove the time complexity of your algorithm as a function of n.

O(n logn) because of the sort as the computing itself would take O(n).

Problem 5 (10 points)

Suppose you were to drive from **Riyadh** to **Jeddah** along a straight one way highway. Your gas tank, when full, holds enough gas to travel ***M*** miles, and you have a map that gives distances between gas stations along the route. Let $d_1 < d_2 < \ldots < d_n$ be the locations of all the gas stations along the route where d_i is the distance from **Riyadh** to the gas station. You can assume that the distance between neighboring gas stations is at most ***M*** miles.

Your goal is to make as few gas stops as possible along the way;

a- Give the most efficient algorithm you can find to determine at which gas stations you should stop.

```
trip(d[1...n], m)
 S[] <- Ø
  dis <- M
 for i=1 to n do
  if(dis - d[i] < 0 )
    S[i] <- d[i-1]
    dis = M
   else
   dis = dis - d[i]
return S
```

b- Prove the time complexity of your algorithm as a function of n.

O(n) as . . - - - -

c- Which programming design technique did you use?

greedy ..

bc there is a greed proprity which is the Least no. of stops.